RELATÓRIO – EP4

Tabela de Símbolos

Pedro Gigeck Freire - 10737136

Implementação do Exercício:

As implementações foram completamente baseadas no que vimos em sala de aula (listas e árvores principalmente).

Para cada estrutura, foi feita uma *struct* para guardar as palavras com suas ocorrências.

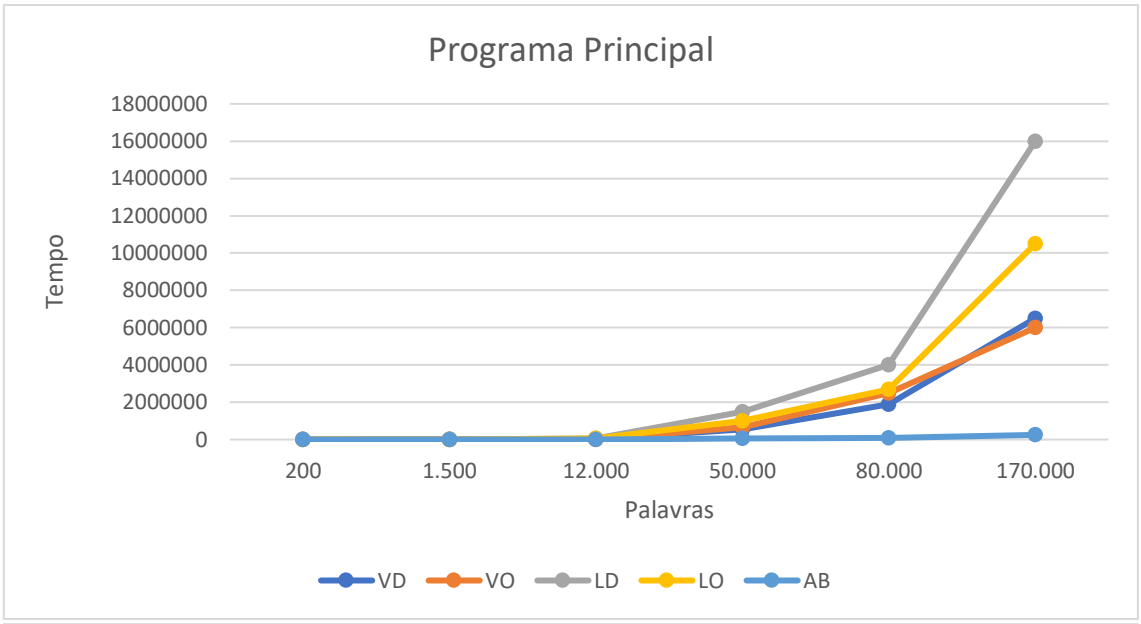
Na impressão da saída, coloquei todas as estruturas em um vetor, facilitando bastante a ordenação, como só é feito uma vez, a criação desse vetor contribui muito pouco para o tempo consumido da execução do programa.

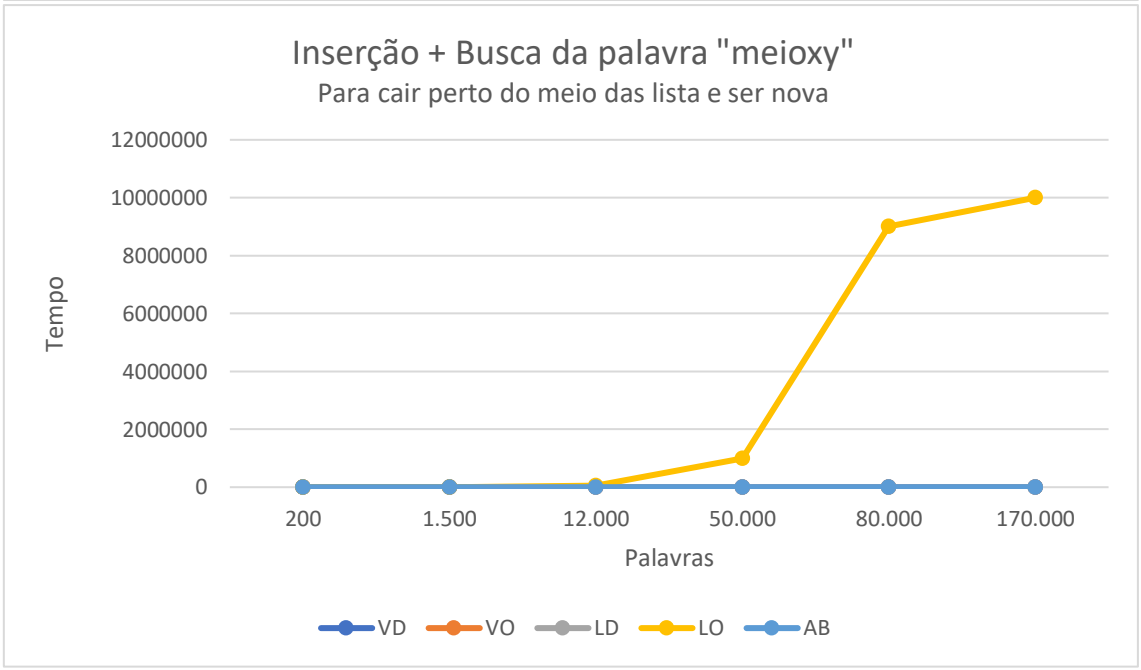
A ordenação do vetor foi feita com a ideia de um *quicksort*, feito com a versão da função *separa* do *Sedgewick*, visto em aula (pegando o pivô no final etc), com uma versão do código para cada ordenação desejada (alfabética e por ocorrências).

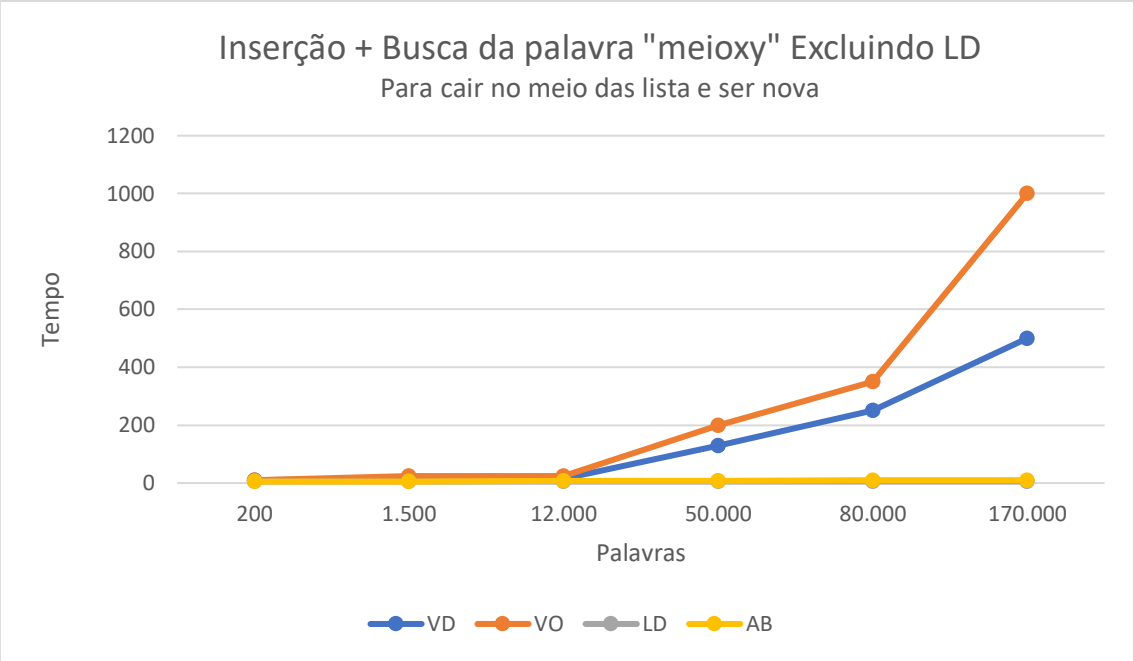
TESTES:

Os testes maiores foram realizados com textos de Hamlet (50000) Lusiadas (80000) e um anexo de Lusiadas e Bras Cubas (170000), ainda testei com textos em francês e inglês, mas a principal diferença se deu pela acentuação, que acabou gerando mais palavras nos textos latinos, em geral, ignorando acentos, o tempo foi muito similar.

Primeiramente, vamos ver o comportamento de cada estrutura quando executado o programa completo: *(1000000 = 1 segundo)*


Programa Principal
18000000
16000000
14000000
12000000
10000000
8000000
6000000
4000000
2000000
0
Tempo
200
1.500
12.000
50.000
80.000
170.000
Palavras
VD
VO
LD
LO
AB


Inserção + Busca da palavra "meioxy"
Para cair perto do meio das lista e ser nova
12000000
10000000
8000000
6000000
4000000
2000000
0
Tempo
200
1.500
12.000
50.000
80.000
170.000
Palavras
VD
VO
LD
LO
AB



CONCLUSÕES:

Os vetores têm comportamentos muito similares, uma vez que o desordenado demora muito para fazer a busca, mas insere rapidamente, enquanto o ordenado consegue realizar buscas binárias, mas tem que empurrar todos os elementos para o final, ficando em muita desvantagem na inserção, como vimos no gráfico.

Sobre as listas, vimos que seu uso não é interessante nesse caso, com a lista ordenada, não conseguimos fazer uma busca binária para obter vantagem, igualmente no caso desordenado, onde a busca deve percorrer toda a lista.

Podemos reparar, no primeiro gráfico, que a complexidade dos vetores e das listas são muito similares, diferindo apenas de algumas constantes. Pois cada uma dessas estruturas tem algum trecho de complexidade $O(n^2)$, que é como os gráficos se assemelham.

A consideração mais importante a fazer é que a Árvore de Busca Binária é uma estrutura muito eficiente! Sua estrutura agiliza as buscas e inserções de modo impressionante. E é muito improvável um pior caso, com textos com muitas palavras. Mesmos que em casos menos favorecidos, a altura cresce pouco.